MUSIC GENRE CLASSIFICATION: NEAR-REALTIME VS SEQUENTIAL APPROACH

Scott Zhang | Huaping Gu | Rongbin Li

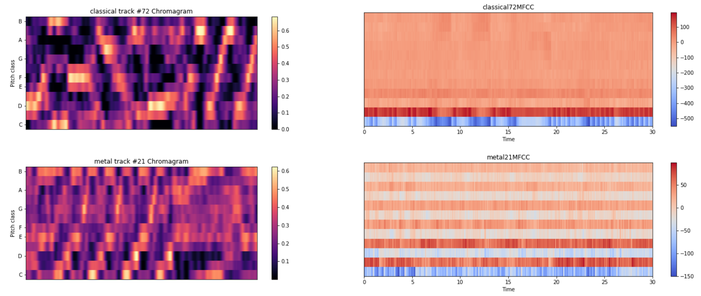
Stanford University

Abstract

In this project, we use a bunch of algorithms we learnt from the class CS229 to classify music into different Genres. To get reliable performance bench-marking, we use the industry famous GTZAN Genre Collection dataset. We start with calculating the classifications with several traditional classic algorithms, then compare with some modern deep learning algorithms with heuristics layer design and data augmentation. We propose an unique near real-time music genre classification solution, by listening a music 0.5 seconds we can identify its genre with 64% accuracy. We conclude that RNN is powerful in classifying a longer track, while even short time MFCC features can already get decent classification accuracy. Using the mean and covariance of MFCC as an augmented feature set is inspired by Million Song Dataset (corresponding to the field section_timbre), and its subset in UCI repository.

Dataset and Methodology

We are using the GTZAN Genre Collection dataset. GTZAN has total 10 different genres, each containing 100 audio clips that are 30 seconds long. We use librosa Python library to extract features from the wave files. Librosa will return a 2D array. e.g. if we use Mel-frequency Cepstral Coefficients (MFCC) we will get one (12×1293) array for a 30 seconds 220 Hz music with hop-length=512. x coordinate is time and y coordinate is 1 of the 12 coefficients. In our project, we will use two librosa methods to extract the raw data from the wave file, chromagram and MFCC, of the same shape. Bellow are plotted output for two genre 2D arrays.



We realized that in the GTZAN samples, some musics are little longer than the expected 30 seconds long which makes the extracted array length > 1293, so before we load data to the next step algorithms we do some array slicing.

Feature Extraction Algorithms

We first use librosa API load the music into an 12x1293 2D music, then we $mean$ on the 1293 fields into a 1×12 1D. Then we combine these features with the ground truth (extract from the folder structure) and genre_id, get one $(14 \times n)$ 2D array. Finally for n samples, we get $x \times 14$ wd array. We use these features for out bench-marking tests.

Result: Extracted song features (n $\times 14$)
initialization;
while *i < n* **do**
  librosa$_i[12 \times 1293] =$ $load\_by\_librosa()$
  feature$_i[1 \times 12] = mean_by_time()$
  features$_i[1 \times 14] = fill_GT_genre_id$
end
return features;

Ground Truth Label	Genre ID	12 features

Algorithm 1: Benchmark Features

In our basic bench-marking step, we use left algorithm $n \times 14$ 2D array. But in the enhanced feature extraction, we append extra 78 co-variance features, totally we get a $(n \times 92)$ 2D array.We use these extended features in all the other tests, including CNN and RNN.

Result: Extracted song features (n$\times 92$)
initialization;
while *i < n* **do**
  librosa$_i[12 \times 1293] =$ $load\_by\_librosa()$
  feature$_i[1 \times 12] = mean_by_time()$
  features$_i[1 \times 14] = fill_GT_genre_id$
  features$_i[1 \times 92] =$ $ext\_cov\_78(feature_i)$
end
return features;

Ground Truth Label	Genre ID	12 features	78 Co_variance features

Algorithm 2: Extended Features

Benchmark Results

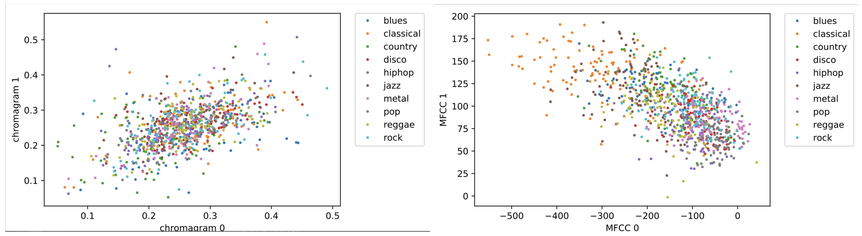
We used logistic regression We used 4 Logistic regression algorithms with multi-nomial option on, all of them we use the l_2 error.

- "newton-cg":newton method
- "lbfgs":Broyden–Fletcher–Goldfarb–Shanno
- "sag":Stochastic Average Gradient
- "saga":Gradient Method With Non-Strongly Convex Composite
- SVC:Support Vector Classification
- LinearSVC: Linear Support Vector Machine

For each set of the librosa features on bellow algoriths, we got following results:

Features/Algorithms	MFCC	Chromagram
Newton-cg	51.5%	19%
lbfgs	51%	19%
sag	49%	19%
saga	50%	19%
SVC	42.5%	30%
LinearSVC	35.5%	24.5%

From the benchmark results above, we found that MFCC performs much better than the Chromagram. Based on the definitions of MFCC and Chromagram, it is not surprise to see this results. In the benchmark data extraction step, we did $mean()$ on the sample's dataset which did lose some of the information.



For Chromagram extractions, since it is related to the twelve different pitch classes, which are very common and used across all different genre. And after we do the $mean()$, we cannot see the differences between genre. (left plot). But for MFCC, it is related to the musical instrument, intuitively instrument type does related to the music genre. (right plot)

Dataset Standardization

We do pre-processing by standardizing the extracted features, placing the input features in the same scaled space. We sometimes add $L2$ regularization while training, to avoid overfitting. On average, with the standardization and regularization, we can see some percentage improvement on the final test accuracy on all algorithms above, especially the SVM algorithms on MFCC.

Features/Algorithms	MFCC	Chromagram
Newton-cg	53.4%	25.8%
lbfgs	53.2%	26.0%
sag	51.0%	25.8%
saga	51.4%	25.8%
SVC	61.5%	28%
LinearSVC	50.0%	25.0%

Classical Model on Segmented Dataset

We segmented the 30 seconds music into multiple small parts. For example, if we divide each song into 30 segments (each has 1-second long), we would totally get $30,000$ samples, and for each segmented sample, the size is $1 \times (90 + 1)$ (90 features plus 1 label). We run classical model (multiclass LogisticRegression, SVC, LinearSVC) on the segmented dataset, get the results on the right. From the results, we can see that, the best test acc we can get for segmentated data is about 70%, but there is still obviously overfitting. To get a better result, we need to consider time series and apply a sequential approach (see sections below).

Models	number of Seg	Training Acc.	Test Acc.
LR	1	82.00%	71.50%
LR	3	78.16%	68.16%
LR	10	70.97%	64.60%
LR	30	63.39%	59.91%
SVC	1	90.12%	70.50%
SVC	3	92.20%	71.66%
SVC	10	90.77%	69.25%
SVC	30	86.61%	66.28%
LinearSVC	1	93.37%	71.50%
LinearSVC	3	82.95%	64.50%
LinearSVC	10	71.46%	63.4%
LinearSVC	30	63.73%	59.03%

Fully connected Neural Network

For NN we use the de-faco industry standard framework *Tensorflow.keras*. For all the datasets used bellow are all MFCC with 90 features $((12 + 78))$.
We start with bellow NN network settings

L1 Input Dim	L1 activation	L1 regularizer	L2 Input Dim	L2 activation
90	relu	$l_2(0.01)$	10	softmax

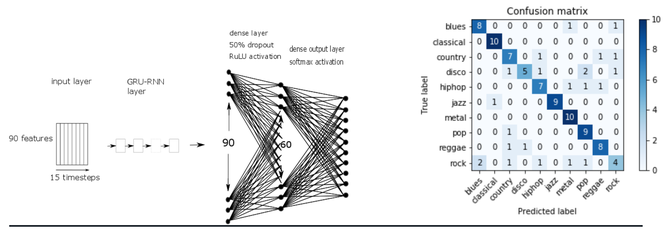
We run 200 epochs with 800 batch size, with various segmentatin intervals:

Batch Size	Epoch	Optimizer	segments	Train Accuracy	Test Accuracy
800	200	rmsprop	1	70.88%	63.00%
800	200	rmsprop	10	74.14%	67.00%
800	200	rmsprop	20	69.95%	65.25%
800	200	rmsprop	30	67.34%	62.70%

Recurrent Neural Network

We implemented RNN using

- An input layer of shape (15, 90), i.e. segmenting audio into timesteps.
- A layer of GRU with 90 output units, with 50% dropout rate and 10% recurrent dropout rate,
- A fully connected layer of 60 units, with ReLU activation, 50% dropout rate, and L2 regularization,
- An output layer of 10 units, with softmax activation.



segments	GRU #	Dense #	Dropout %	Train Accuracy	Test Accuracy
15	90	60	50	84.57%	77%
30	60	40	20	95.71%	72%
30	90	60	50	83.71%	75%
60	90	60	50	84.14%	70%

Conclusions

- MFCC features (average + covariance) are relatively constant over time, so that even a short 0.5 second clip of music can be classified into 10 genres with more than 65% accuracy consistently.
- Taking MFCC average and covariance over longer period of time will improve the accuracy up to 70%, even for classical algorithms such as SVM classifier.
- Treating MFCC average and covariance as a time series, a GRU-RNN can boost the test accuracy to near 80%, proving that the time progression of MFCC indeed has predictive power in classifying genre.

Acknowledgements

We really appreciated all the help of the professor Tengyu Ma and Christoher Re, and all the TAs.